REMARKS

Reconsideration and allowance of the above-application are respectfully requested.

Drawings

Fig. 6 has been objected to because "the arrow denoting information flow from box 638 to box 650 contradicts the specification which indicates that information should flow from 638 and from 650 to box 624". This inadvertent error has been corrected.

Fig. 7A has been objected to as failing to include arrows and to label boxes entitled PROCESS NODE and OUTPUT VOICE XML. Fig. 7A has been amended to include arrows where appropriate and to add reference numeral 731 for the PROCESS NODE box and reference numeral 747 for the Output VOICE XML box.

Fig. 8 has been objected to as containing elements without reference numerals. The cell phone has been associated with reference numeral 802 (see, inter alia, Fig. 8), the visual gateway has been associated with reference numeral 880 (see, inter alia, par. 79), the internet has been associated with reference numeral 130 (see, inter alia, Fig. 8), and new reference numeral 90 has been associated with the VXML player (which is now reference in par. 128).

Figs. 12-14 have been objected as containing boxes without references. These boxes have now been numbered and are referenced in the specification.

Priority Claim

Paragraph 1 of the specification has been amended to clarify the priority claim for the current application.

Correction to provide the benefit of all three cited applications is respectfully requested.

Specification

The specification has been objected to as containing computer program code on pages 99-112. These pages have been converted into a computer program listing appendix and proper incorporation by reference has been added to the specification.

35 USC § 112

Claims 35-41 have been rejected under 35 USC § 112, second paragraph as allegedly being indefinite. This rejection is respectfully traversed. Notwithstanding, claim 35 has been amended to clarify that the document contains data for use by an application as opposed to paper or other physical structure with information drawn or written upon it.

35 USC § 103

Claims 1-45 have been rejected as allegedly being unpatentable under 35 USC § 103 over Chung in view of Gergic.

Claim 1 defines a method comprising the steps of receiving a document in a structured language which includes tags associated with portions of the document, using the tags to provide a speech mark-up language version from the document, and using the tags, and the same document, to provide a visual mark-up language version from the document.

Chung describes a system for converting a HTML document to speech including an HTML parser, an HTML to speech converter (HTS) control parser, a tag converter, a text normalizer and a TTS converter. The HTML parser receives data of an HTML formatted document and parses out HTML text tags that structure

content text and control used for translating the received data into sound (see, inter alia, Chung abstract).

Gergic describes the possibility of multi-modal applications that are able to operate on multiple channels (see, inter alia, Gergic paragraphs 37, 38). While Gergic may provide an arrangement that utilizes speech mark-up language documents and visual mark-up language documents, it does not describe generating both a speech mark-up language version of the document and a visual mark-up language version of the document from the tags within the document.

The skilled artisan would not have been motivated to combine the teachings of Chung and Gergic to result in the subject matter of claim 1. Chung does not contemplate providing a speech mark-up language document and a visual mark-up language document from the tags within a document. Rather, it describes the parsing of HTML text tags that are used for translating the data into sound.

Gergic is focused on reusable dialog components to simplify building speech-based applications by associating various parameters with the dialog components. Although Gergic describes the use of VoiceXML, there is no direction or other suggestion for a skilled artisan to use tags to provide both speech mark-up language and visual mark-up language documents.

As neither of Chung or Gergic describe using tags to provide both speech mark-up language documents and visual mark-up language documents, there is no suggestion for the skilled artisan to (i) combine Chung and Gergic; and (ii) change the fundamental techniques taught by Chung and Gergic to practice the method of claim 1.

Accordingly, claim 1 and the claims dependent thereon should be allowable.

The rejection of claims 3, 9, 23, 43, 44, and 45 is further traversed. The Gergic passage cited in the office action is an ambiguous reference to the selection of a menu item from a menu list in connection with the parameterization of template components. Gergic states that it may be compatible with different modalities but it is silent with regard to switching between such modalities. In particular, it does not disclose switching from either a voice or visual mark-up language document to the other document. Therefore, there is no suggestion within Gergic that would have lead the skilled artisan to practice the subject matter of any of claims 3, 9, 23, 43, 44, and 45.

Claim 3, for example, defines switching from one of the document versions to the other of the document versions. Claim 9 defines receiving a document with a SWITCH tag that allows a user to switch content from one of the mark-up versions to the other mark-up version. Claim 23 defines allowing the content to be browsed partly in visual form and partly in voice form. Claim 43 defines accepting a request from a portable phone to switch a mark-up language which is being displayed, and switching the mark-up language between visual mark-up language and speech mark-up language. Claim 44 defines accepting a request from a portable phone to display additional language, and displaying both speech mark-up language and visual mark-up language at the same time. None of these aspects are disclosed or suggested by Gergic.

Therefore, the skilled artisan would not have adapted the menu list of Gergic in combination with Chung to result in a method having any of the specifically defined steps of claims 3, 9, 23, 43, and 44.

Moreover, claim 45 defines allowing a user to browse a document in either of voice XML or compact visual XML, keeping

track of the position of browsing in a current language, and allowing switching from a current language to another language at a position related to the position of browsing. Gergic fails to disclose both the tracking and switching aspects of this claim. As a result, the skilled artisan would not have been motivated to combine Chung and Gergis to resulting the subject matter of claim 45.

Accordingly, 3, 9, 23, 43, 44, and 45 as well as those dependent thereon should be allowable.

The rejection of claims 5 and 6 is further traversed.

Claim 5 defines allowing the user to browse a document in visual mark-up language, and accepting a command to provide additional content in the speech mark-up language version at the same time as providing the visual mark-up language version. Claim 6 defines allowing browsing a document in a speech mark-up language and accepting a command to provide additional content in visual markup language at the same time.

Chung and Gergis do not contemplate using tags to provide both speech mark-up language and visual mark-up language documents. Moreover, these reference do not suggest to selectively provide additional content in a speech mark-up language document or a visual mark-up language document. Therefore, these references would not have been combined by the skilled artisan to result in the subject matter of claims 5 and 6 which relate to both speech mark-up language and visual mark-up language documents and the provision of additional content in the version of the document that is not being browsed.

Accordingly, claims 5 and 6 should be allowable.

The rejection of claims 10-13 is further traversed. Claim 10 clarifies that the step of determining a need for visual mark-up language comprises determining if an initiating terminal has visual mark-up language capability. Claim 11 clarifies that

the step of determining if the initiating terminal has visual mark-up capability comprises sending a message to the initiating terminal and determining if the initiating terminal completes registration based on the message. Claim 12 defines that the initiating terminal is a handheld phone. Claim 13 clarifies that the step of determining if the initiating terminal has visual mark-up capability comprises informing the terminal to access a predetermined website which requires visual mark-up capabilities, and determining if the terminal has access to the website.

Neither Chung nor Gergis describe the determination of whether an initiating terminal has visual mark-up language capability. This is a threshold requirement for the subject matter defined by claims 10-13 Moreover, with regard to claim 11, neither of the cited references contemplate sending a message to an initiating terminal to determine capabilities. In addition, with regard to claim 13, neither of the references suggest informing the terminal to access a predetermined website.

Accordingly, claims 10, 11, 12, and 13 and those dependent thereon should be allowable.

The rejection of claim 14 is respectfully traversed. Neither of Chung nor Gergis even suggest utilizing short message services nor similar technologies. As a result, the skilled artisan would not have combined these references to determine if the if the initiating terminal has short message service capabilities as claimed.

Accordingly, claim 14 should be allowable.

The rejection of claim 24 is respectfully traversed. Claim 24 defines dividing source documents into multiple parts, and converting each of the multiple parts into either or both of visual mark-up language and/or voice mark-up language. Neither

Chung nor Gergic suggest such a step in which the source document is divided.

Accordingly, claim 24 should be allowable.

The rejection of claim 25 is respectfully traversed. Claim 25 defines a system comprising an information storage unit which stores a document in structured language that includes tags associated with portions of the document, and a conversion server which allows converting the tags to provide both information in a voice mark-up language and information in a visual mark-up language, based on the same document. Neither Chung nor Gerbis suggest converting tags within a document to provide information both in a voice mark-up language and information in a visual mark-up language. Nor is there any suggestion to combine these references in any fashion that would result in the subject matter of claim 25.

Accordingly, claim 25 and those dependent thereon should be allowable.

The rejection of claim 35 is respectfully traversed. Claim 35 defines a document containing data for use by an application that comprises a structured format including text attributes and tag attributes, at least one of the text attributes being convertible into both a voice mark-up language and a visual mark-up language, and providing information which can be used in both of the voice mark-up and visual mark-up languages.

Chung only contemplates providing documents having information in only a visual mark-up language. In addition, Gergis contemplates using multiple documents each providing information in a single mark-up language. As a result, there is no motivation for the skilled artisan to combine the references to practice the subject matter of claim 35. Moreover, neither of these references also contemplate providing information which can be used in both of the voice mark-up and visual mark-up

languages. Therefore, even if the cited references were combined, the skilled artisan would not have modified their teachings to result in the subject matter of claim 35.

Accordingly, claims 35 and those dependent thereon should be allowable.

Concluding Comments

It is believed that all of the pending claims have been addressed in this paper. However, failure to address a specific rejection, issue or comment, does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above are not intended to be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

Applicant asks that all claims be allowed. Please apply any charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 1/24/05

Scott C. Harris
Reg. No. 32,030

Fish & Richardson P.C.
PTO Customer Number: **20985**
12390 El Camino Real
San Diego, CA 92130
Telephone: (858) 678-5070
Facsimile: (858) 678-5099

APPENDIX A

```
/*
* Function : convert
*
* Input    : filename, document base
*
* Return   : None
*
* Purpose  : parses the input wml file and converts it into vxml file.
*
*/
    public void convert(String fileName,String base)
    {
      try {
      Document doc;
        Vector problems = new Vector();

        documentBase = base;

      try {
            VXMLErrorHandler errorhandler = new
VXMLErrorHandler(problems);

          DocumentBuilderFactory docBuilderFactory =
DocumentBuilderFactory.newInstance();
          DocumentBuilder docBuilder =
docBuilderFactory.newDocumentBuilder();

          doc = docBuilder.parse (new File (fileName));

            TraverseNode(doc);

            if (problems.size() > 0){
                Enumeration enum = problems.elements();
                while(enum.hasMoreElements())
                    out.write((String)enum.nextElement());
            }
      } catch (SAXParseException err) {
          out.write ("** Parsing error"
            + ", line " + err.getLineNumber ()
            + ", uri " + err.getSystemId ());
          out.write("   " + err.getMessage ());
      } catch (SAXException e) {
          Exception     x = e.getException ();

          ((x == null) ? e : x).printStackTrace ();
      } catch (Throwable t) {
          t.printStackTrace ();
      }
      } catch (Exception err) {
          err.printStackTrace ();
        }

    }
```

APPENDIX B

EXEMPLARY WML TO VOICEXML CONVERSION

WML to VoiceXML Mapping Table

[0001] The following set of WML tags may be converted to VoiceXML tags of analogous function in accordance with Table B1 below.

TABLE B1

WML Tag	VoiceXML Tag
Access	*Access*
Card	*form*
Head	*Head*
Meta	*meta*
Wml	*Vxml*
Br	*Break*
P	*Block*
Exit	*Disconnect*
A	*Link*
Go	*Goto*
Input	*Field*
Option	*Choice*
Select	*Menu*

Mapping of Individual WML Elements to Blocks of VoiceXML Elements

[0002] In an exemplary embodiment a VoiceXML-based tag and any required ancillary grammar is directly substituted for the corresponding WML-based tag in accordance with Table A1. In cases where direct mapping from a WML-based tag to a VoiceXML tag would introduce inaccuracies into the conversion process, additional processing is required to accurately map the information from the WML-based tag into

a VoiceXML-based grammatical structure comprised of multiple VoiceXML elements. For example, the following exemplary block of VoiceXML elements may be utilized to emulate the functionality of the to the WML-based *Template* tag in the voice domain.

WML-Based Template Element

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<template>
        <do type="options" label="DONE">
                <go href="test.wml"/>
        </do>
  </template>
  <card>
         <p align="left">Test</p>
<select name="newsitem">
         <option onpick="test1.wml">Test1 </option>
             <option onpick="test2.wml">Test2</option>
              </select>
  </card>
</wml>
```

Corresponding Block of VoiceXML Elements

```
<?xml version="1.0" ?>
<vxml version="1.0">
   <link next="test.vxml">
     <grammar>
       [
         (DONE)
       ]
     </grammar>
   </link>
   <menu>
      <prompt>Please say test1 or test2</prompt>
<choice next="test1.vxml"> test1 </choice>
<choice next="test2.vxml"> test2 </choice>
   </menu>
 </vxml>
```

Example of Conversion of Actual WML Code to VoiceXML Code

Exemplary WML Code

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Deck Source: "http://wap.cnet.com" -->
<!-- DISCLAIMER: This source was generated from parsed binary WML
content. -->
<!-- This representation of the deck contents does not
necessarily preserve -->
<!-- original whitespace or accurately decode any CDATA Section
contents, -->
<!-- but otherwise is an accurate representation of the original
deck contents -->
<!-- as determined from its WBXML encoding.  If a precise
representation is required, -->
```

```
<!-- then use the "Element Tree" or, if available, the "Original
Source" view. -->

<wml>
  <head>
    <meta http-equiv="Cache-Control" content="must-revalidate"/>
    <meta http-equiv="Expires" content="Tue, 01 Jan 1980 1:00:00
GMT"/>
    <meta http-equiv="Cache-Control" content="max-age=0"/>
  </head>

  <card title="Top Tech News">
    <p align="left">
      CNET News.com
    </p>
    <p mode="nowrap">
      <select name="categoryId" ivalue="1">
        <option onpick="/wap/news/briefs/0,10870,0-1002-903-1-
0,00.wml">Latest News Briefs</option>
        <option onpick="/wap/news/0,10716,0-1002-
901,00.wml">Latest News Headlines</option>
        <option onpick="/wap/news/0,10716,0-1007-901,00.wml">E-
Business</option>
        <option onpick="/wap/news/0,10716,0-1004-
901,00.wml">Communications</option>
        <option onpick="/wap/news/0,10716,0-1005-
901,00.wml">Entertainment and Media</option>
        <option onpick="/wap/news/0,10716,0-1006-
901,00.wml">Personal Technology</option>
        <option onpick="/wap/news/0,10716,0-1003-
901,00.wml">Enterprise Computing</option>
      </select>
    </p>
  </card>

</wml>
```

Corresponding VoiceXML code

```
<?xml version="1.0"?>
<vxml version="1.0">
<head> <meta/> <meta/> <meta/>
</head>
<form>
<block>
<prompt>CNET News.com</prompt>
</block>
<block>
<grammar>
[  ( latest news briefs )  ( latest news headlines )  ( e-
business )  ( communic
ations )  ( entertainment and media )  ( personal technology )  (
enterprise com
puting ) ]
</grammar>
<goto next="#categoryId" />
</block>
</form>
```

```
<menu  id="categoryId"  >
<property name="inputmodes" value="dtmf" />
<prompt>Please Say <enumerate/>
</prompt>
<choice dtmf="0"   next="http://server:port/Convert.jsp?url=
http://wap.cnet.com/wap/news/briefs/0,10870,0-1002-903-1-
0,00.wml"> Latest News Briefs </choice>
<choice dtmf="1"   next="http:// server:port
/Convert.jsp?url=http://wap.cnet.com/wap/news/0,10716,0-1002-
901,00.wml"> Latest News Headlines </choice>
<choice dtmf="2"   next="http:// server:port
/Convert.jsp?url=http://wap.cnet.com/wap/news/0,10716,0-1007-
901,00.wml"> E-Business </choice>
<choice dtmf="3"   next="http:// server:port
/Convert.jsp?url=http://wap.cnet.com/wap/news/0,10716,0-1004-
901,00.wml"> Communications </choice>
<choice dtmf="4"   next="http:// server:port/Convert.jsp?url=
http://wap.cnet.com/wap/news/0,10716,0-1005-901,00.wml">
Entertainment and Media </choice>
<choice dtmf="5"   next="http:// server:port /Convert.jsp?url=
http://wap.cnet.com/wap/news/0,10716,0-1006-901,00.wml"> Personal
Technology </choice>
<choice dtmf="6"   next="http:// server:port /Convert.jsp?url=
http://wap.cnet.com/wap/news/0,10716,0-1003-901,00.wml">
Enterprise Computing </choice>
<default>
<reprompt/>
</default>
</menu>
</vxml>


<! END OF CONVERSION >
```

APPENDIX C

```
/*
* Function : TraverseNode
*
* Input    : Node
*
* Return   : None
*
* Purpose  : Traverse's the Dom tree node by node and converts the
*            tag and attributes into equivalent vxml tags and
attributes.
*
*/
    void TraverseNode(Node el){

        StringBuffer buffer = new StringBuffer();

        if (el == null)
            return;
        int type = el.getNodeType();

        switch (type){
            case Node.ATTRIBUTE_NODE: {
                    break;
                }
            case Node.CDATA_SECTION_NODE: {
                    buffer.append("<![CDATA[");
                    buffer.append(el.getNodeValue());
                    buffer.append("]]>");
                    writeBuffer(buffer);
                    break;
                }
            case Node.DOCUMENT_FRAGMENT_NODE: {
                    break;
                }
            case Node.DOCUMENT_NODE: {
                    TraverseNode(((Document)el).getDocumentElement());
                    break;
                }
            case Node.DOCUMENT_TYPE_NODE : {
                    break;
                }
            case Node.COMMENT_NODE: {
                    break;
                }
            case Node.ELEMENT_NODE: {
                    if (el.getNodeName().equals("select")){
                        processMenu(el);
                    }else if (el.getNodeName().equals("a")){
                        processA(el);
                    } else {
                    buffer.append("<");
                    buffer.append(ConvertTag(el.getNodeName()));
                    NamedNodeMap nm = el.getAttributes();
                    if (first){
                        buffer.append(" version=\"1.0\"");
```

```
                        first=false;
                    }
                    int len = (nm != null) ? nm.getLength() : 0;
                    for (int j =0; j < len; j++){
                        Attr attr = (Attr)nm.item(j);

buffer.append(ConvertAtr(el.getNodeName(),attr.getNodeName(),attr.getNo
deValue()));
                    }
                    NodeList nl = el.getChildNodes();
                    if ((nl == null) ||
                        ((len = nl.getLength()) < 1)){
                       buffer.append("/>");
                       writeBuffer(buffer);
                    }else{
                       buffer.append(">");
                       writeBuffer(buffer);
                       for (int j=0; j < len; j++)
                           TraverseNode(nl.item(j));
                       buffer.append("</");
                       buffer.append(ConvertTag(el.getNodeName()));
                       buffer.append(">");
                       writeBuffer(buffer);
                    }
                    }
                    break;
                }
            case Node.ENTITY_REFERENCE_NODE : {
                    NodeList nl = el.getChildNodes();
                    if (nl != null){
                        int len = nl.getLength();
                        for (int j=0; j < len; j++)
                            TraverseNode(nl.item(j));
                    }
                    break;
                }
            case Node.NOTATION_NODE: {
                    break;
                }
            case Node.PROCESSING_INSTRUCTION_NODE: {
                    buffer.append("<?");
                    buffer.append(ConvertTag(el.getNodeName()));
                    String data = el.getNodeValue();
                    if ( data != null && data.length() > 0 ) {
                       buffer.append(" ");
                       buffer.append(data);
                    }
                    buffer.append(" ?>");
                    writeBuffer(buffer);
                    break;
                }
            case Node.TEXT_NODE: {
                    if (!el.getNodeValue().trim().equals("")){
                        try {

out.write("<prompt>"+el.getNodeValue().trim()+"</prompt>\n");
                        }catch (Exception e){
```

```
                    e.printStackTrace();
                }
            }
            break;
        }
    }
}
/*
```

APPENDIX D

```
/*
* Function : ConvertTag
*
* Input    : wpa tag
*
* Return   : equivalent vxml tag
*
* Purpose  : converts a wml tag to vxml tag using the
WMLTagResourceBundle.
*
*/
    String ConvertTag(String wapelement){
        ResourceBundle rbd = new WMLTagResourceBundle();
        try {
            return rbd.getString(wapelement);
        }catch (MissingResourceException e){
            return "";
        }
    }
/*
* Function : ConvertAtr
*
* Input    : wap tag, wap attribute, attribute value
*
* Return   : equivalent vxml attribute with it's value.
*
* Purpose  : converts the combination of tag+attribute of wml to a vxml
*            attribute using WMLAtrResourceBundle.
*
*/

    String ConvertAtr(String wapelement,String wapattrib,String val){

        ResourceBundle rbd = new WMLAtrResourceBundle();
        String tempStr="";
        String searchTag;
        searchTag =wapelement.trim()+"-"+wapattrib.trim();
        try {
          tempStr += " ";
            String convTag = rbd.getString(searchTag);
            tempStr += convTag;
            if (convTag.equalsIgnoreCase("next"))
              tempStr += "=\""+server+"?url="+documentBase;
            else
              tempStr += "=\"";
          tempStr += val;
          tempStr += "\"";
            return tempStr;
        }catch (MissingResourceException e){
            return "";
        }
    }
/*
* Function : processMenu
*
```

```
* Input     : Node
*
* Return    : None
*
* Purpose   : process a menu node. it converts a select list into an
*             equivalent menu in vxml.
*
*/
    private void processMenu(Node el){
        try {
        StringBuffer mnuString = new StringBuffer();
        StringBuffer mnu = new StringBuffer();
        String menuName ="NONAME";
        int dtmfId = 0;
        StringBuffer mnuGrammar = new StringBuffer();
        Vector menuItem = new Vector();

        mnu.append("<"+ConvertTag(el.getNodeName()));
        NamedNodeMap nm = el.getAttributes();
        int len = (nm != null) ? nm.getLength() : 0;
        for (int j =0; j < len; j++){
            Attr attr = (Attr)nm.item(j);
            if (attr.getNodeName().equals("name")){
                menuName=attr.getNodeValue();
            }
            mnu.append(" " +
ConvertAtr(el.getNodeName(),attr.getNodeName(),attr.getNodeValue()));
        }
        mnu.append(">\n");
        mnu.append("<property name=\"inputmodes\" value=\"dtmf\"
/>\n");
        NodeList nl = el.getChildNodes();
        len = nl.getLength();

        for (int j=0; j < len; j++){
            Node el1 = nl.item(j);
            int type = el1.getNodeType();
            switch (type){
                case Node.ELEMENT_NODE: {
                    mnuString.append("<"+ConvertTag(el1.getNodeName())
+" dtmf=\"" + dtmfId++ +"\" ");
                    NamedNodeMap nm1 = el1.getAttributes();
                    int len2 = (nm1 != null) ? nm1.getLength() : 0;
                    for (int l =0; l < len2; l++){
                        Attr attr1 = (Attr)nm1.item(l);
                        mnuString.append(" " +
ConvertAtr(el1.getNodeName(),attr1.getNodeName(),attr1.getNodeValue()))
;
                    }
                    mnuString.append(">\n");
                    NodeList nl1 = el1.getChildNodes();
                    int len1 = nl1.getLength();
                    for (int k=0; k < len1; k++){
                        Node el2 = nl1.item(k);
                        switch (el2.getNodeType()){
                            case Node.TEXT_NODE: {
```

```
                                    if
(!el2.getNodeValue().trim().equals("")){

mnuString.append(el2.getNodeValue()+"\n");

menuItem.addElement(el2.getNodeValue());
                                    }
                                   }
                                   break;
                              }
                         }

mnuString.append("</"+ConvertTag(el1.getNodeName())+">\n");
                         break;
                    }
               }

          }
          mnuString.append("<default>\n<reprompt/>\n</default>\n");
          mnuString.append("</"+ConvertTag(el.getNodeName())+">\n");
          mnu.append("<prompt>Please Say <enumerate/>");
          mnu.append("\n</prompt>");
          mnu.append("\n"+mnuString.toString());

          mnuGrammar.append("<grammar>\n[ ");
          for(int i=0; i< menuItem.size(); i++){
              mnuGrammar.append(" ( " + menuItem.elementAt(i) + " ) ");
          }
          mnuGrammar.append("]\n</grammar>\n");

          out.write(mnuGrammar.toString().toLowerCase());
          out.write("\n<goto next=\"#" + menuName +"\".
/>\n</block>\n</form>\n");
          out.write(mnu.toString());
          out.write("<form>\n<block>\n");
          }catch (Exception e){
              e.printStackTrace();
          }
     }
/*
* Function : processA
*
* Input    : link Node
*
* Return   : None
*
* Purpose  : converts an <A> i.e. link element into an equivalent for
*            vxml.
*
*/
     private void processA(Node el){
          try {

          StringBuffer linkString = new StringBuffer();
          StringBuffer link = new StringBuffer();
          StringBuffer nextStr = new StringBuffer();
          StringBuffer promptStr = new StringBuffer();
```

```
        String fieldName ="NONAME"+field_id++;
        int dtmfId = 0;
        StringBuffer linkGrammar = new StringBuffer();

        NamedNodeMap nm = el.getAttributes();
        int len = (nm != null) ? nm.getLength() : 0;

        linkGrammar.append("<grammar> [(next) (dtmf-1) (dtmf-2) ");
        for (int j =0; j < len; j++){
            Attr attr = (Attr)nm.item(j);
            if (attr.getNodeName().equals("href")){
            nextStr.append("<goto "
+ConvertAtr(el.getNodeName(),attr.getNodeName(),attr.getNodeValue())
+"/>\n");
            }
        }

        linkString.append("<field name=\""+fieldName+"\">\n");

        NodeList nl = el.getChildNodes();
        len = nl.getLength();
        link.append("<filled>\n");
        for (int j=0; j < len; j++){
            Node el1 = nl.item(j);
            int type = el1.getNodeType();
            switch (type){
                 case Node.TEXT_NODE: {
                      if (!el1.getNodeValue().trim().equals("")){
                          promptStr.append("<prompt> Please Say Next or
"+el1.getNodeValue()+"</prompt>");

linkGrammar.append("("+el1.getNodeValue().toLowerCase()+")");
                          link.append("<if cond=\""+fieldName+" ==
'"+el1.getNodeValue()+"' || "+fieldName+" =='dtmf-1'\">\n");
                          link.append(nextStr);
                          link.append("<else/>\n");
                          link.append("<prompt>Next
Article</prompt>\n");
                          link.append("</if>\n");
                      }
                 }
                 break;
            }

        }
        linkGrammar.append("]</grammar>\n");
        link.append("</filled>\n");
        linkString.append(linkGrammar);
        linkString.append(promptStr);
        linkString.append(link);
        linkString.append("</field>\n");
        out.write("</block>\n");
        out.write(linkString.toString());
        out.write("<block>\n");
        }catch (Exception e){
            e.printStackTrace();
        }
```

```
    }
/*
* Function : writeBuffer
*
* Input    : buffer String
*
* Return   : None
*
* Purpose  : print the buffer to PrintWriter.
*
*/

    void writeBuffer(StringBuffer buffer){

        try {
            if (!buffer.toString().trim().equals("")){
                out.write(buffer.toString());
                out.write("\n");
            }
        }catch (Exception e){
            e.printStackTrace();
        }

        buffer.delete(0,buffer.length());
    }
      }
```

10469052.doc